APPLICATION NOTE

AP-272

June 1986

The 5C060 Unification of a CHMOS System

J. R. DONNELL
PROGRAMMABLE LOGIC APPLICATIONS
INTEL CORPORATION

Order Number: 292009-003

INTRODUCTION

From an outside glance, the world of computers and microprocessors seems filled with dedicated ICs that fulfill a variety of system needs. Upon closer inspection we find that designers must still reach into their bag of random logic to link together all of the parts of the system. It seems a shame to stuff a board full of high powered peripherals and still have portions of that board wasted on decoders, latches, and other miscellaneous random logic.

True, programmable logic has been around a long time. But that logic is somewhat rigid in form, one time programmable, and can also double as space heaters. These devices are totally unacceptable for a CMOS system. What is needed is a flexible PLA architecture, erasability for prototyping, and CMOS for low power. In addition, for this particular application the device must perform from static operation to 10 MHz.

OBJECTIVE

This application note covers the design of three separate circuits for Intel's CHMOS Design Kit. The functions performed by the 5C060 are: Memory decoding, wait state generation, and the power down circuitry for the 80C88 system clock.

MEMORY DECODING

The system in question supports one 32K bank of EPROM memory, and four banks of 4K static RAM. Figure 1 shows the memory map of this system. Address lines A19, A13, and A12 will be used to decode the address space. PWR_DWN and S2_MIO serve as enables. In addition, to avoid data bus contention signals memory read (MRDC) and advanced memory write (AMWC) are decoded along with the address lines for RAM chip selects. This is necessary for devices without output enables (OE) on multiplexed address/data busses.

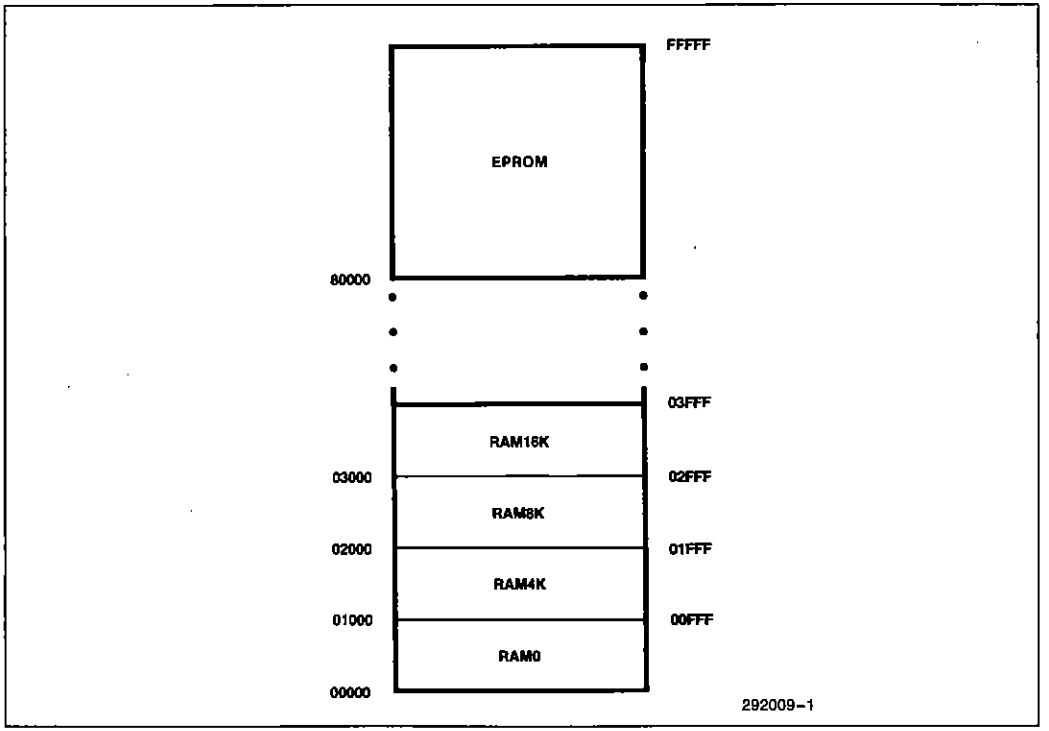

Figure 1. 80C88 Memory Map

Figure 2 shows a discrete implementation of the chip select decoding logic.

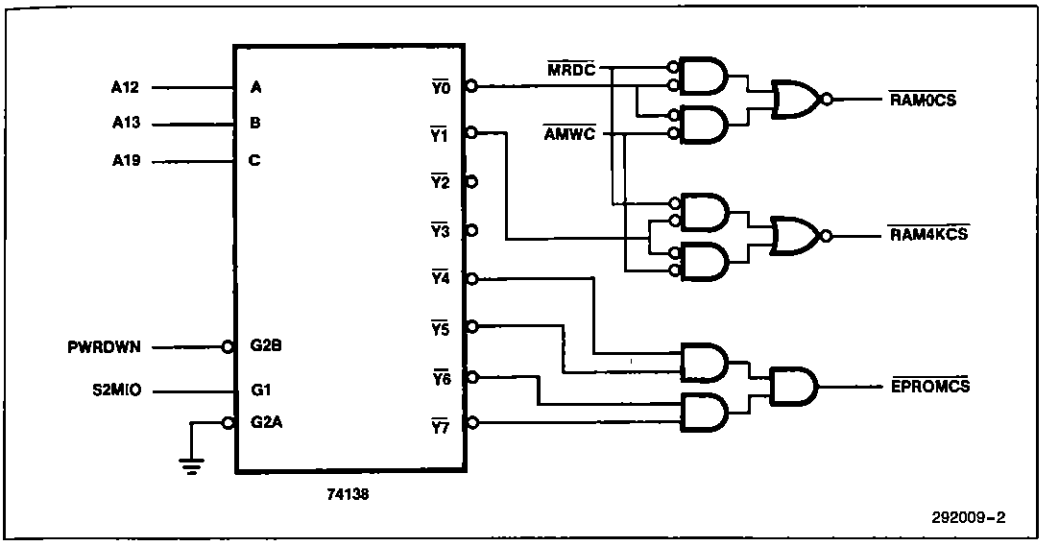

Figure 2. Discrete Decoding Logic Solution

Several options for entering this design are available through Intel's Programmable Logic Development System (iPLDS). (For a more complete description of iPLDS the reader is referred to the iPLDS data sheet.) The design entry vehicle chosen for this application note is the Logic Builder. (Logic Builder is an interactive netlist method of design entry especially suited to Boolean equation entry and entry from existing schematics.) Several reasons are behind this decision. First, the Logic Builder software is included in iPLDS. In addition, Logic Builder entry is very fast, the designer may choose either netlist entry or Boolean equations, and finally, the Logic Builder software makes additions and corrections of design very easy.

Using Logic Builder, the first step for this design is to determine the equations for the 3 to 8 decoder shown in Figure 2. These equations are simply the decoding of the address lines ANDed with the enable signal. Equations 0 thru 8 implement the decoding function of Figure 2.

```
/Y0 = /A19*/A13*/A12*ENABLE;          (0)
/Y1 = /A19*/A13*A12*ENABLE;           (1)
/Y2 = /A19*A13*/A12*ENABLE;           (2)
/Y3 = /A19*A13*A12*ENABLE;            (3)
/Y4 = A19*/A13*/A12*ENABLE;           (4)
/Y5 = A19*/A13*A12*ENABLE;            (5)
/Y6 = A19*A13*/A12*ENABLE;            (6)
/Y7 = A19*A13*A12*ENABLE;             (7)
ENABLE = /PWRDWN*S2MIO;               (8)
```

Armed with this knowledge it becomes trivial to enter the circuit of Figure 2 into Logic Builder. Included in the Appendix is the Advanced Design File (ADF) created by Logic Builder for this circuit (ADF-1). Typically the ADF would now be submitted to the Logic Optimizing Compiler (LOC) for Boolean minimization and design fitting. In this case we have used only a small portion if the logic available in the 5C060 so let us continue with the wait state generator and power down circuitry.

Power Down

Since this design is based on the 80C88 we can actually stop the system clock for extended periods of time and power back up as if nothing had occurred. The circuit to achieve this power down is shown in Figure 3.

As long as the PWRDWN signal is low the 82C84 clock output is OR'ed with a logical zero from the PWRDWN flip-flop. As a result the 82C84 drives the 80C88 system clock. If PWRDWN goes HIGH, the rising edge of the next 82C84 clock will set the output of the PWRDWN flip-flop HIGH inhibiting the fall of the next clock cycle. The 80C88 system clock will remain HIGH until PWRDWN goes LOW and the PWRDWN flip-flop is clocked from the 82C84 clock. Using this configuration we avoid partial clock cycles for the 80C88 system clock.

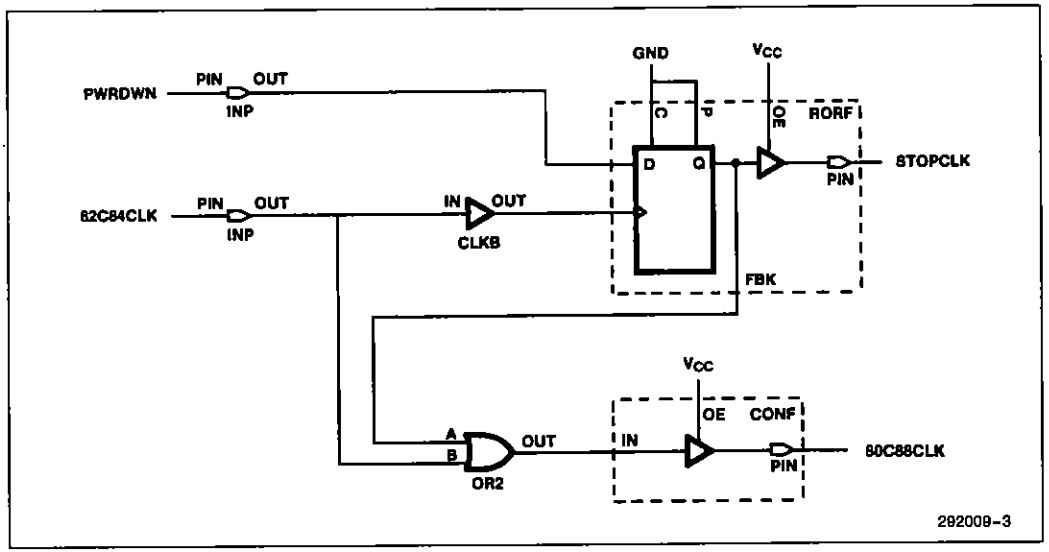

Figure 3. 80C88 Power Down Circuit

Again, entering this circuit into Logic Builder is trivial. In fact it can be added directly to the decoder circuit shown above. The ADF file for this addition is shown in the appendix under ADF-2.

Wait States

The majority of memory and peripheral devices which fail to operate at the maximum CPU frequency typically do not require more than one wait state. The circuit shown in Figure 4 is an example of a simple wait state generator. The circuit operation is as follows. Given that a memory location requiring a wait state has been selected, ALE in conjunction with /WAITCS will clear the flip-flop—driving the 82C84RDY line high low. The 82C84 samples the RDY line during T2 of the 80C88 bus cycle, and in this case detects a wait state. The rising edge of T2 then clocks the 82C84RDY line high thereby inserting only one wait state.

Once again, adding this circuit to the existing decoder and power down design is simple. The final ADF file is given in the appendix under ADF-3. Once the final design has been completed the ADF is submitted to the Logic Optimizing Compiler. LOC compiles the design, performs Boolean minimization, and fits the design into the target EPLD. In addition, LOC produces two files. The JEDEC programming file, the Logic Equation File (LEF), and the Utilization Report. These are also included in the appendix for each step in this design process.

LOC FILES

The JEDEC File

The JEDEC file is analogous to the object code file that is used to program EPROMs. This file is used by the Logic Programming Software (LPS) to program Intel's EPLDs.

The LEF File

The LEF file is an optional file produced by the compiler. The LEF file contains the minimized Boolean equations which resulted from the original ADF. Some interesting points can be raised concerning the LEF file. Looking at LEF-3, first recall that the EPROM chip select was a function of A19, A13, A12, and the enable signals. It turns out that after minimization the EPROM chip select depends only on A19 and the enable signals (/PWRDWN and S2MIO). This is shown in the LEF file. One other point, the initial wait state circuitry employed a JK flip-flop. The compiler automatically minimized this circuit into a D-type flip-flop with feedback achieving the same functionality.

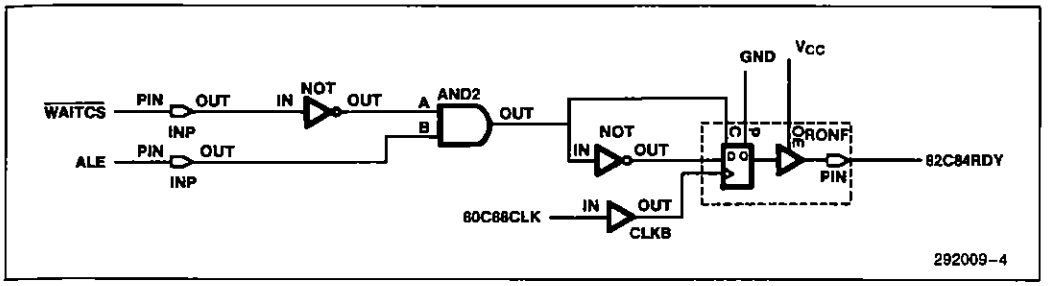

Figure 4. Single Wait State Generator for the 80C88

The Utilization Report

Finally, the Utilization Report contains the pin-out for the design, information about the architectural layout of the design, and a percent utilization for pins, macrocells, and product terms. Examining the utilization report for this design we find that two of the sixteen macrocells are still available. We could therefore add more functionality in the same 24 pin package. Possible additions would be more memory decoding, invalid memory detection, additional wait state generators, etc. One point should be raised: The circuitry designed in this applications note is relatively simple compared to the complex logic functions that could be implemented in the 5C060.

SUMMARY

The designs shown in this applications note are typical requirements of any microprocessor system. The 5C060 provided a single chip solution to bind together the primary elements of that system. Few other types of programmable logic could implement the same logic in a single package. None could do it in CMOS erasable logic. The 5C060 has room for more.

APPENDIX

ADF-1

```
JR Donnell
Intel
January 31, 1986
5C060
0
5C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
LR Version 3.0, Baseline 17x, 9/26/85
PART: 5C060
INPUTS: A19,A13,A12,PWRDWN,S2MIO,AMWC,MRDC
OUTPUTS: RAM0CS,RAM4KCS,RAM8KCS,RAM16KCS,RPROMCS
NETWORK:
RAM0CS = CONF (RAM0CS,VCC)
RAM4KCS = CONF (RAM4KCS,VCC)
RAM8KCS = CONF (RAM8KCS,VCC)
RAM16KCS = CONF (RAM16KCS,VCC)
RPROMCS = CONF (EPROMCS,VCC)
A19 = INP (A19)
A13 = INP (A13)
A12 = INP (A12)
PWRDWN = INP (PWRDWN)
S2MIO = INP (S2MIO)
MRDC = INP (MRDC)
AMWC = INP (AMWC)
EQUATIONS:
RAM8KCS = //(/MRDC*Y2
        + /AMWC*Y2):
RAM16KCS = //(/MRDC*Y3
         + /AMWC*Y3):
RPROMCS = //(Y7
        + /Y6
        + /Y5
        + /Y4):
Y7 = /(A19*A13*A12*RNABLR):
Y6 = /(A19*A13*/A12*ENABLE):
Y5 = /(A19*/A13*A12*RNABLR):
Y4 = /(A19*/A13*/A12*ENABLE):
RNABLR = /PWRDWN*S2MIO:
Y3 = /(/A19*A13*A12*ENABLE):
Y2 = /(/A19*A13*/A12*RNABLR):
RAM4KCS = /(/MRDC*Y1
        + /AMWC*Y1):
Y1 = /(/A19*/A13*A12*ENABLE):
RAM0CS = /(/MRDC*Y0
       + /AMWC*Y0):
Y0 = /(/A19*/A13*/A12*RNABLR):
END$
```

292009-5

ADF-2

```
JR Bonnell
Intel
January 31, 1986
8C060
0
8C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
Plus power down circuit
LB Version 3.0, Baseline 17x, 9/26/86
PART: 8C060
INPUTS: A19,A13,A12,PWRDWN,S2MIO,AMWC,MRDC,82C84CLK
OUTPUTS: RAM0CS,RAM4KCS,RAM8KCS,RAM16KCS,RPROMCS,STOPCLK,80C88CLK
NETWORK:
RAM0CS = CONF (RAM0CS,VCC)
RAM4KCS = CONF (RAM4KCS,VCC)
RAM8KCS = CONF (RAM8KCS,VCC)
RAM16KCS = CONF (RAM16KCS,VCC)
RPROMCS = CONF (EPROMCS,VCC)
STOPCLK,STOPCLKF = RORF (PWRDWN,82C84CLKB,GND,GND,VCC)
80C88CLK = CONF (80C88CLK,VCC)
PWRDWN = INP (PWRDWN)
82C84CLKB = CLKB (82C84CLK)
80C88CLK = OR (STOPCLKF,82C84CLK)
82C84CLK = INP (82C84CLK)
A19 = INP (A19)
A13 = INP (A13)
A12 = INP (A12)
S2MIO = INP (S2MIO)
MRDC = INP (MRDC)
AMWC = INP (AMWC)
EQUATIONS:
RAM0CS = /(/MRDC*Y0
         + /AMWC*Y0);
RAM4KCS = /(/MRDC*Y1
         + /AMWC*Y1);
RAM8KCS = /(/MRDC*Y2
         + /AMWC*Y2);
RAM16KCS = /(/MRDC*Y3
         + /AMWC*Y3);
RPROMCS = /(/Y7
         + /Y6
         + /Y5
         + /Y4);
Y0 = /(/A19*/A13*/A12*ENABLE);
Y1 = /(/A19*/A13*A12*ENABLE);
Y2 = /(/A19*A13*/A12*ENABLE);
Y3 = /(/A19*A13*A12*ENABLE);
Y7 = /(A19*A13*A12*ENABLE);
Y6 = /(A19*A13*/A12*ENABLE);
Y5 = /(A19*/A13*A12*ENABLE);
Y4 = /(A19*/A13*/A12*ENABLE);
ENABLE = /PWRDWN*S2MIO;
END$
```

292009-6

ADF-3

```
JR Donnell
Intel
January 31, 1986
5C060
0
5C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
Plus power down circuit
Plus wait state circuit
LB Version 3.0, Baseline 17x, 9/26/85
PART: 5C060
INPUTS: A19,A13,A12,PWRDWN,S2MIO,AMWC,MRDC,82C84CLK,ALE,WAITCS
OUTPUTS: RAM0CS,RAM4KCS,RAM8KCS,RAM16KCS,EPROMCS,STOPCLK,80CARCLK,82CR4RDY
NETWORK:
RAM0CS = CONF (RAM0CS,VCC)
RAM4KCS = CONF (RAM4KCS,VCC)
RAM8KCS = CONF (RAM8KCS,VCC)
RAM16KCS = CONF (RAM16KCS,VCC)
EPROMCS = CONF (EPROMCS,VCC)
STOPCLK,STOPCLKF = RORF (PWRDWN,82CR4CLKB,GND,GND,VCC)
80CARCLK,80C88CLKF = COIF (80C88CLK,VCC)
82CR4RDY = RONF (82CR4RDYD,80CARCLKB,82CR4RDYC,GND,VCC)
PWRDWN = INP (PWRDWN)
82CR4CLKB = CLKB (82CR4CLK)
80CARCLK = OR (STOPCLKF,82C84CLK)
82CR4CLK = INP (82CR4CLK)
A19 = INP (A19)
A13 = INP (A13)
A12 = INP (A12)
S2MIO = INP (S2MIO)
MRDC = INP (MRDC)
AMWC = INP (AMWC)
80CARCLKB = CLKB (80C88CLKF)
WAITCS = INP (WAITCS)
ALE = INP (ALE)
EQUATIONS:
RAM0CS = /(/MRDC*Y0
         + /AMWC*Y0);
RAM4KCS = /(/MRDC*Y1
         + /AMWC*Y1);
RAM8KCS = /(/MRDC*Y2
         + /AMWC*Y2);
RAM16KCS = /(/MRDC*Y3
         + /AMWC*Y3);
EPROMCS = /(/Y7
         + /Y6
         + /Y5
         + /Y4);
Y0 = /(/A19*/A13*/A12*ENABLE);
Y1 = /(/A19*/A13*A12*ENABLE);
Y2 = /(/A19*A13*/A12*ENABLE);
Y3 = /(/A19*A13*A12*ENABLE);
Y7 = /(A19*A13*A12*ENABLE);
Y6 = /(A19*A13*/A12*ENABLE);
Y5 = /(A19*/A13*A12*ENABLE);
Y4 = /(A19*/A13*/A12*ENABLE);
ENABLE = /PWRDWN*S2MIO;
82CR4RDYD = /82CR4RDYC;
82C84RDYC = /WAITCS*ALE;
END$
```

292009-7

```
JR Donnell
Intel
January 31, 1986                    LEF-3
5C060
0
5C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
Plus power down circuit
Plus wait state circuit
LB Version 3.0, Bangalore 17x, 9/26/85
PART:
        5C060
INPUTS:
        A19, A13, A12, PWRDWN, S2MIO, AMWC, MRDC, 82C84CLK, ALE, WAITCS
OUTPUTS:
        RAM0CS, RAM4KCS, RAM8KCS, RAM16KCS, EPROMCS, STOPCLK, 80C88CLK,
        82C84RDY
NETWORK:
        A19 = INP(A19)
        A13 = INP(A13)
        A12 = INP(A12)
        PWRDWN = INP(PWRDWN)
        S2MIO = INP(S2MIO)
        AMWC = INP(AMWC)
        MRDC = INP(MRDC)
        82C84CLK = INP(82C84CLK)
        ALE = INP(ALE)
        WAITCS = INP(WAITCS)
        RAM0CS = CONF(RAM0CS, VCC)
        RAM4KCS = CONF(RAM4KCS, VCC)
        RAM8KCS = CONF(RAM8KCS, VCC)
        RAM16KCS = CONF(RAM16KCS, VCC)
        EPROMCS = CONF(EPROMCS, VCC)
        ..SG000D = CLKB(82C84CLKB)
        STOPCLK, STOPCLKF = RORF(PWRDWN, ..SG000D, GND, GND, VCC)
        80C88CLK, 80C88CLKF = COIF(80C88CLK, VCC)
        ..SG001D = CLKB(80C88CLKB)
        82C84RDY = RONF(82C84RDYD, ..SG001D, 82C84RDYC, GND, VCC)
EQUATIONS:
        82C84RDYC = WAITCS' * ALE;

        ..SG001D = 80C88CLKF;

        82C84RDYD = (WAITCS' * ALE)';

        80C88CLK = (STOPCLKF' * 82C84CLK')';

        ..SG000D = 82C84CLK;

        EPROMCS = (A19 * PWRDWN' * S2MIO)';

        RAM16KCS = MRDC * AMWC
                 + A19' * A13 * A12 * PWRDWN' * S2MIO;

        RAM8KCS = MRDC * AMWC
                + A19' * A13 * A12' * PWRDWN' * S2MIO;

        RAM4KCS = MRDC * AMWC
                + A19' * A13' * A12 * PWRDWN' * S2MIO;

        RAM0CS = MRDC * AMWC
               + A19' * A13' * A12' * PWRDWN' * S2MIO;

END$
```

292009-8

RPT-3

```
Logic Optimizing Compiler Utilization Report

 ***** Design implemented successfully

JR Donnell
Intel
January 31, 1986
6C060
0
6C060
Decoder for 80C88 system - 16K RAM and upper 512K EPROM
Plus power down circuit
Plus wait state circuit
LR Version 3.0. Baseline 17x. 9/26/85

              5C060
            - - - - -
       GND -: 1   24:- Vcc
    PWRDWN -: 2   23:- A19
       GND -: 3   22:- STOPCLK
       GND -: 4   21:- R2CR4RDY
    WAITCS -: 5   20:- 80C88CLK
       ALE -: 6   19:- RPROMCS
  R2CR4CLK -: 7   18:- RAM16KCS
      MRDC -: 8   17:- RAM8KCS
      AMWC -: 9   16:- RAM4KCS
     S2MTO -:10   15:- RAM0CS
       A12 -:11   14:- A13 .
       GND -:12   13:- GND
            - - - - -

**INPUTS**

                                                 :          Feeds:
      Name  Pin  Resource   MCell #  PTerms     :  MCells     OR  Clear  Clock

    PWRDWN    2       INP         -         -          1      -      -      -
                                                       4
                                                       5
                                                       6
                                                       7
                                                       8

    WAITCS    5       INP        11      0/ 8          2      -     -2      -

       ALE    6       INP        12      0/ 8          2      -      2      -

  R2CR4CLK    7       INP        13      0/ 8          3      -      -      1

      MRDC    8       INP        14      0/ 8          5      -      -      -
                                                       6
                                                       7
                                                       8

      AMWC    9       INP        15      0/ 8          5      -      -      -
                                                       6
                                                       7
                                                       8

     S2MTO   10       INP        16      0/ 8          4      -      -      -
                                                       5
```

292009-9

```
                                                          6
                                                          7
                                                          8

        A12   11        TNP          -          -        5        -       -       -
                                                          6
                                                          7
                                                          8

        A13   14        TNP          -          -        5        -       -       -
                                                          6
                                                          7
                                                          8

        A19   23        TNP          -          -        4        -       -       -
                                                          5
                                                          6
                                                          7
                                                          8


**OUTPUTS**

                                                          Feeds:
      Name  Pin  Resource    MCell #    PTerms  : MCells     OR   Clear   Clock

    RAM0CS   15       CONF          8      2/ 8          -       -       -       -
    RAM4KCS  16       CONF          7      2/ 8          -       -       -       -
    RAM8KCS  17       CONF          6      2/ 8          -       -       -       -
   RAM16KCS  18       CONF          5      2/ 8          -       -       -       -
    RPROMCS  19       CONF          4      1/ 8          -       -       -       -
   ROCARCLK  20       COTF          3      1/ 8          -       -       -       2
   82C84RDY  21      RONFA          2      1/ 8          -       -       -       -
    STOPCLK  22      RORFA          1      1/ 8          3       -       -       -

**UNUSED RESOURCES**

      Name  Pin  Resource      MCell     PTerms

         -    1         -          -          -
         -    3         -          9          8
         -    4         -         10          8
         -   13         -          -          -

**PART UTILIZATION**

81%      Pins
87%      MacroCells
9%       Pterms
```

292009-10